Forecasting of Maximum and Minimum Temperature in Mandalay by Evolving Artificial Neural Networks Using Genetic Algorithms

Khaing Shwe Wutyi
University of Computer Studies Mandalay
wutyi.pan@gmail.com

Abstract

Forecasting of weather is very popular in nowadays. But forecasting the future from the observed past is very difficult. There are several forecasting methods for weather data. Among them, evolving artificial neural networks are suitable for weather time series forecasting because of their abilities to learn and adapt a new situation by recognizing new patterns in previous data. However, ANNs present some drawbacks such as over fitting and long time processing. Using ANNs together with genetic algorithm comes to solutions of these problems. Genetic artificial neural networks (GANNs) can give optimal forecasting result from the observed past. In order to provide more effective ANNs, the proposed system use cascade back propagation instead of back propagation method. Weather parameters (attributes) such as rain fall (precipitation), humidity, wind force, dew point, sea level, wind direction will also be used to forecast maximum and minimum temperature.

Keywords: Artificial Neural Networks (ANN), Genetic Algorithms (GA), Time Series (TS), Mean Square Error(MSE), Specific Mean Square Error(SMSE), Multi Layer Perceptron(MLP)

1. Introduction

In order to acquire knowledge, it is interesting to know what the future will look like, i.e. forecast the future from the observed past. Time series forecasting is an essential research field due to its effectiveness in human life. It is a discipline that finds each day more applications in areas like planning, management, production, maintenance, and control of industrial process, economy and weather forecasting.

The forecasting task can be performed by several techniques as Statistical methods [2], and others based on Computational Intelligence like Immune Systems [3] and Artificial Neural Networks (ANNs) [4].

ANNs provide a methodology for solving many types of nonlinear problems that are difficult to solve by traditional techniques. Most time series processes often exhibit temporal and spatial variability, and are suffered by issues of nonlinearity of physical processes, conflicting spatial and temporal scale and uncertainty in parameter estimates. The ANNs have capability to extract the relationship between the inputs and outputs of a process, without the physics being explicitly provided. Thus, these properties of ANNs are well suited to the problem of time series forecasting.

This contribution reports the methodology to carry out the automatic design of ANN that tackles the forecasting of a referenced set of time series. The task will consist of forecasting several time series, not all of them with the same ANN, but an automatic method will be used to

obtain a different ANN to forecast each time series [4].

Two different steps [5], will be done to get an ANN to forecast each time series. The first step will consist of setting the kind of ANN that will solve the forecasting task (Multilayer Perceptron), and the learning algorithm used (Back propagation).

In the second step the design of the ANN will be done, setting the parameter values of the ANN, i.e. number of input nodes, number of hidden nodes, learning rate for BP and finally all connections weights. These parameters are established by carrying out a search process performed by a Genetic Algorithm (GA).

2. Time Series and ANN

Before using an ANN to forecast, it has to be designed, i.e. establishing the suitable value for each freedom degree of the ANN [8] (kind of net, number of input nodes, number of outputs neurons, number of hidden layer, number of hidden neurons, the connections from one node to another , connection weights, etc).

The problem of forecasting time series with ANN is considered as modeling the relationship of the value of the element in time *"t"* (due to the net will only have one output neuron) and the values of previous elements of the time series (*t-1, t-2,.., t-k*) to obtain a function as it is shown in (1):

$$a_t = f(a_{t-1}, a_{t-2}, \ldots, a_{t-k}) \qquad (1)$$

2.1. Artificial Neural Network Architecture

There is no known rule that points out to what kind of *ANN* one must use. The choice is done by looking at the data, to the problem, and using intuition or experience [5] [14]. One's approach for time series forecasting supposes the use of feed forward *ANNs*.

The decision was to use feed forward *ANNs* fully connected, with bias, without shortcut connections, and with one hidden layer, which makes possible to bound at a certain level the *ANN* parameters, in order to reduce the vectorial search space. The initial *ANN* weights were randomly generated within the range of $[\frac{-2}{z}; \frac{2}{z}]$ for a node with *z* inputs. Standard back-propagation [5] was used for the *ANN* training.

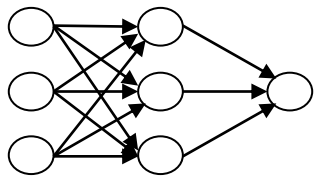

Figure 1. Multilayer Perceptron (MLP) with One-Hidden Layer (Fully-Connected)

The ANN topology will be represented by for an ANN with L_i input nodes, L_h hidden nodes and L_o output nodes [14]. In this work L_o is always equal to one. Empirical tests indicated that, with the same data, ANNs with one output node gave better forecasts than ANNs with *n* output nodes, even for *n* prediction steps. The ANN receives *n* previous TS values, for *n* input nodes, and outputs the forecasted value. The training cases were built as follows:

$$\begin{aligned} &x_1, x_2, \ldots, x_k && \rightarrow x_{k+1} \\ &x_2, x_3, \ldots, x_{k+1} && \rightarrow x_{k+2} \\ &\ldots\ldots && \rightarrow \ldots\ldots \\ &x_{p-k}, \ldots, x_{p-1} && \rightarrow x_p \end{aligned}$$

where $x_1, \ldots, x_p$ represent the *TS* and *k* the number of the ANN's inputs. The cases are trained in this order due to the temporal nature of the data. The trained *ANN* can make short or long term predictions:

$f_{1,p}$ = *output 1*(x_{p-k}, … ,x_p)
$f_{2,p}$ = *output 1*(x_{p-k+1}, … ,x_p,$f_{1,p}$)
$f_{3,p}$ = *output 1*(x_{p-k+2}, … ,x_p,$f_{1,p}$,$f_{2,p}$)
……..
$f_{n,p}$ = *output 1*($f_{n-k,p}$,… ,$f_{n-1,p}$)

where output 1(x_1,…,x_k) caters for the output function of the ANN, and $f_{i,j}$ the forecast in the *j* period to *i* periods ahead of *j[5]*.

2.2. Specific Mean Square Error (SMSE)

One problem that arises with back-propagation training is over fitting: after some training epochs the *ANN* starts to lose generalization. To overcome this situation, early stopping was considered [14]. The training data was divided into the training set (for the *ANN* learning) and the validation set (to test the *ANN* generalization capacity). By default the validation set is 10% of the training data, since the use of the last (recent) values in the training set improves the forecast.

The best *ANN* architecture for short and long term forecasts tends to differ. On long term forecasts *ANNs* seem to be more affected by over fitting. Thus a special validation test, *Specific Mean Square Error (SMSE)*, that takes into consideration the number of forecasts ahead, was implemented:

$$MSE_{n,j} = \frac{\left((f_{1,j}-x_{j+1})^2+(f_{2,j}-x_{j+2})^2+\cdots+(f_{n,j}-x_{j+n})^2\right)}{n} \quad (2)$$

$$SMSE_n = \frac{\sum_{j=p-n-k}^{p-n} MSE_{n,j}}{k} \quad (3)$$

where *p* is the last period of the *TS*, *k* the number of validation cases and *n* the number of forecasting steps. $MSE_{n,j}$ is the error measure for *n* forecasting steps from *j* , and *n* is an user defined parameter.

2.3. Genetic Algorithms

A genetic algorithm is a computer algorithm that searches for a good solution to a problem among a large number of possible solutions. These search algorithms are based on the mechanism of natural selection and natural genetics. Genetic algorithms maintain a population of some feasible solutions for a given problem. This population undergoes evolution in a form of natural selection and natural genetics. All information required for creation of appearance and behavioral features of living organism is contained in its chromosomes. Reproduction generally involves two parents, and the chromosomes of the offspring are generated from portions chromosomes taken from the parents. In this ways, the offspring inherit a combination of characteristics from their parents. Gas attempts to use a similar method of inheritance to solve various problems [25].

2.4. Five Components of GA

Five components required by GA are defined by the system designer. In this system, system designer defined a way of encoding solution to the problem as binary form.

1. A way of encoding solution to the problem on chromosome or candidate solution.
2. An evaluation function which returns a rating for each chromosome given to it.
3. A way of initializing the population of chromosomes.
4. Operators that may be applied to parents when they reproduce to alter their genetic composition. Standard operators are mutation and crossover.
5. Parameter settings for the algorithm. [26]

```
while evolution do
(1)  select one rsp. two parents              selection
(2)  generate one offspring                   mutation rsp. crossover
(3)  tune offspring and                       learning
     evaluate its fitness                     & evaluation
(4)  insert offspring in population           survival of the fittest
     and delete the last population element
```

Figure 2.Genetic Algorithm Skeleton

3. ANN Design with GA

In this work an initial approach to design Artificial Neural Networks to forecast time series is tackle, and the automatic process to design is carried out by a Genetic Algorithm. A key issue for these kinds of approaches is what information is included in the chromosome that represents an Artificial Neural Network.

There are two principal ideas about this question: first, the chromosome contains information about parameters of the topology, architecture, learning parameters, etc. of the Artificial Neural Network, i.e. Direct Encoding Scheme; second, the chromosome contains the necessary information so that a constructive method gives rise to an Artificial Neural Network topology (or architecture), i.e. Indirect Encoding Scheme.

3.1. Learning Pattern Set

In order to obtain a single ANN to forecast time series values, an initial step has to be done with the original values of the time series, i.e. normalize the data. And once the ANN gives the resulting values, the inverse process is carried out in order to evaluate the forecasting carried out by the ANN. This step is important as the ANN works with real values number between 0 and 1 as input values. Furthermore, the time series known values will be transformed into a pattern set, depending on the k inputs nodes of each ANN generated in search process carry out by GA. Therefore, each pattern consists in:

- "*k*" inputs values, that correspond to "*k*" normalized previous values of period *t*: $a_{t-1}, a_{t-2}, \ldots, a_{t-k}$.
- One output value, that corresponds to normalized time series value of period *t*.

This patterns set will be used to train and validate each ANN generated during the GA or DE execution. So patterns set will be split into two subsets, train and validation. The first 90% from the total patterns set will generate the train patterns subset, and the validation subset will be obtained from the rest of the complete patterns set.

3.1.1. ANN Design Carried Out with GA

The problem of designing ANN could be seen as a search problem into the space of all possible ANN. Moreover, that search can be done by a GA [18] using exploitation and exploration.

Therefore there are three crucial issues: i) the solution's space, what information of the net is previously set and what is included into the chromosome; ii) how each solution is codified into a chromosome, i.e. encoding schema; iii) and what is being looked for, translated into the fitness function.

In this approach it has been chosen Multilayer Perceptron (MLP) as computational model due to its approximation capability and inside this group, Full Connected MLP with only a hidden layer and Back Propagation (BP) as learning algorithm, according to [19]. This is because ANN with only one hidden layer are faster to be trained and easier to work than two or more hidden layer MLP.

As it was mentioned before the design of the ANN will be done by setting the parameter values of the ANN. In the case of MLP with only one hidden layer and BP, the parameters are: number of inputs nodes, number of hidden neurons, number of output neurons, (only one, it

is set by the forecasting problem), which is the connection patterns (how the nodes are connected), and the whole set of connection weights (implemented by the seed used to initialize the connection weights as it will be explained later).

For my approach to design ANN to forecast temperature time series, a Direct Encoding Schema for Full Connected MLP has been considered. For this Direct Encoding Scheme the information placed into the chromosome will be: two decimal digits, i.e. two genes, to codify the number of inputs nodes (i); other two for the number of hidden nodes (h); two more for the learning factor (α); and the last ten genes for the initialization seed value of the connection weights (s). This way, the values of "i", "h", "α" and "s" are obtained from the chromosome as it can be seen in (4):

Chrom:

$g_{i1}g_{i2}g_{h1}g_{h2}g_{\alpha1}g_{\alpha2}g_{s1}g_{s2}g_{s3}g_{s4}g_{s5}g_{s6}g_{s7}g_{s8}g_{s9}g_{s10}$ (4)

$0 \leq g_{xy} \leq 9$, x = i, h, α, y = 1, .., 10

$i = max\_inputs \,.\, ((g_{i1} \,.\, 10 + g_{i2})/100)$

$h = max\_hiddens \,.\, ((g_{h1} \,.\, 10 + g_{h2})/100)$

$\alpha = ((g_{\alpha1} \,.\, 10 + g_{\alpha2})/100)$

$s = g_{s1}g_{s2}g_{s3}g_{s4}g_{s5}g_{s6}g_{s7}g_{s8}g_{s9}g_{s10}$

4. The Search Process of GA

The search process (GA) will consist of the following steps:

1. A randomly generated population, i.e. a set of randomly generated chromosomes, is obtained.
2. The phenotypes (i.e. ANN architectures) and fitness value of each individual of the actual generation is obtained. To obtain the phenotype associated to a chromosome and its fitness value:
   2.1 The phenotype of an individual *"i"* of the actual generation is obtained.
   2.2 Then, the train and validation patterns subsets for this neural net "*i*" are obtained from time series data , depending on the number of inputs nodes of the net "*i*", as it was commented above in learning pattern set.
   2.3 The net is trained with BP (The architecture (topology and connections weights set)) of the net when the validation error (i.e. error for validation patterns subset) is minimum during the training process is saved (i.e. early stopping). So this architecture is the final phenotype of the individual.
3. Once that fitness value for whole population is already obtained, the GA operators as Elitism, Selection, Crossover and Mutation are applied in order to generate the population of the next generation, i.e. set of chromosomes.
4. The steps 2 (i.e. 2.1, 2.2, 2.3) and 3 are iteratively executed till a maximum number of generations are reached.

The fitness value for each individual will be then the minimum validation error during the learning process (training of ANN topology), as it can be seen in (5):

*Fitness function = minimum validation error (*5)

The parameters for the GA are: population size, 50; maximum number of generations, 100; percentage of the best individuals that stay unchangeable to the next generation (percentage of elitism), 10%; crossover: parents are split in one point randomly selected, offspring are the mixed of each part from parents; mutation probability will be one divided between the length of the chromosome (*1/length_chrom) = 1/16 ≈ 0.7*), and it will be carried out for each gen of each chromosome.

Once that GA reaches the last generation, the best individual (i.e. ANN) from the last generation is used to forecast the future (and unknown) time series values.

The future unknown values (a_{t+1}) will be forecasted one by one using the *k* previous

known values (a_t, a_{t-1}, …, a_{t-k}). k is the number of inputs of the ANN obtained from the GA execution (i.e. the best ANN from the search process) To forecast several consecutive values (a_{t+1}, a_{t+2},…) every time a new value is forecasted, it will be included in order into the previous known values set of the time series and used to forecast the next one.

5. Experimental Setup

Two different types of temperature time series will be used in this research, maximum temperature and minimum temperature in Mandalay city from the year. Other different weather time series can also be used in this way such as wind direction, wind force, precipitation and humidity.

Table 1. Maximum Average Temperature (Celsius) in Mandalay for the Year 1985 to 1989

Month / Year	Jan	Feb	Mar	April	May	June
1985	28	28	32	39	40	38
1986	29	29	38	40	42	37
1987	32	33	36	42	39	32
1988	30	31	39	44	38	33
1989	29	33	40	45	40	35

Table 2. Minimum Average Temperature (Celsius) in Mandalay for the Year 1985 to 1989

Month / Year	Jan	Feb	Mar	April	May	June
1985	20	20	24	31	32	30
1986	21	21	30	32	34	29
1987	24	25	28	34	31	24
1988	22	27	31	36	30	25
1989	21	26	32	37	32	27

The time series values have to be rescale, into the numerical range value [0, 1], considering not only the known values, but the future values (those to be forecasted). So, the maximum and minimum limits for normalizing (*max4norm*, *min4norm* respectively) cannot be just the maximum (*max*) and minimum (*min*) known time series values. A margin from max and min has to be set if future values were higher or lower than known values already are. This margin will depend on another parameter (*Prct_inc*). In those cases in which the time series is stationary a *Prct_inc* of 10% will be enough, but when the time series is increasing or decreasing *Prct_inc* should be at least of 50%. As it could be forecasted new values for a time series that will rise of a fall, it is needed enough big margin so the new values, obtained as output of ANN, can be into the numerical range [0, 1]. This Equation (6) shows how are obtained max4norm and min4norm.

$$\text{max4norm} = \text{max} + (\text{Prct\_inc} .(\text{max}-\text{min}))$$
$$\text{min4norm} = \text{min} - (\text{Prct\_inc} .(\text{max}-\text{min})) \quad (6)$$

6. Application on GA

Way to forecast two type of temperature time series with GA have been executed five times (200 generations each time) for each time series. For each time series, the average result (and standard deviation) of the two simulations is shown.

To evaluate the error for each method, forecasted values (i.e. test set, not train or validation sets) are compared with real values. Two sorts of errors on forecasted values are used: MSE (mean squared error) and SMAPE (symmetric mean absolute percent error [4]); SMAPE has been used at NN3 and NN5 forecasting competitions.

Table 3. SMAPE and MSE Temperature with GA for 100 Generations

100 Generations		GA
Max Temperature	SMAPE	4.036
	MSE	0.0032
Min Temperature	SMAPE	4.054
	MSE	0.0030

Table 4. SMAPE and MSE Temperature with GA for 200 Generations

200 Generations		GA
Max Temperature	SMAPE	4.024
	MSE	0.0028
Min Temperature	SMAPE	4.019
	MSE	0.0027

7. Conclusion and Further Extension

As it is a totally automatic method, it will not be necessary any previous knowledge from the user. So the user will not have to be an expert in time series, statistics, mathematics or computational intelligence. The user just have to give the time series he wants to forecast and the number of future elements he wants to be forecasted to the system; and this method will give these forecasted values as result to the user.

After running 200 generation, it seems to have good result than 100 generation. So more generation the user tries to run, more good result can get. The user can run until the result of the generation doesn't change.

For further extension, this model can be upgrade to compare with other prediction method such as bayes theorem and differential evolution. And also different time series can be used to forecast for weather prediction. In this system, it is only used two different time series, maximum and minimum temperature time series of Mandalay.

References

[1] Juan Peralta, Xiaodong Li, German Gutierrez, Araceli Sanchis, Time series forecasting by evolving artificial neural networks using genetic algorithms and differential evolution, WCCI 2010 IEEE World Congress on Computational Intelligence July, 18-23, 2010 - CCIB, Barcelona, Spain

[2] Spyros G. Makridakis, Steven C. Wheelwright, Rob J Hyndman. Forecasting: Methods and Applications.

[3] Ian Nunn, Tony White."The application of antigenic search techniques to time series forecasting". Genetic and Evolutionary Computation Conference 2005. ISBN:1-59593-010-8.

[4] Paulo Cortez, José Machado, José Neves. "An evolutionary artificial neural network time series forecasting system". IASTED 1996.

[5] Time Series Forecasting Competition for Neural Networks and Computational Intelligence. http://www.neural-forecastingcompetition. com. Accessed on October 2008.

[6] J. Peralta, G. Gutierrez, A. Sanchis, "ADANN: Automatic Design of Artificial Neural Networks". ARC-FEC 2008 (GECCO 2008). ISBN 978-1-60558-131-6.

[7] Zhang, G.; Patuwo, B.E. & Hu, M.Y. Forecasting with artificial neural networks: The state of the art International Journal of Forecasting, 1998, 14, 35-62.

[8] Haykin, S. Simon & Schuster (ed.) Neural Networks. A Comprehensive Foundation Prentice Hall, 1999.

[9] Crone, S. F. Stepwise Selection of Artificial Neural Networks Models for Time Series Prediction Journal of Intelligent Systems, Department of Management Science Lancaster University Management School Lancaster, United Kingdom, 2005.

[10] T. Ash. Dynamic Node Creation in Backpropagation Networks ICS Report 8901, The Institute for Cognitive Science, University of California, San Diego (Saiensu-sh, 1988), 1988.

[11] D.B. Fogel, Fogel L.J. and Porto V.W. Evolving Neural Network, Biological Cybernetics, 63, 487-493, 1990.

[12] Gruau F. "Genetic Synthesis of Boolean Neural Networks with a Cell Rewriting Developmental Process". Proceedings of COGANN-92 International

Workshop on Combinations of Genetic Algorithms and Neural Networks, pp. 55-74, IEEE Computer Society Press, 1992.
[13] Yao, X. and Lin, Y. A new evolutionary system for evolving artificial neural networks, Transactions on Neural Networks, 8(3): 694-713, 1997.
[14] Kitano, H.: Designing Neural Networks using Genetic Algorithms with Graph Generation System, Complex Systems, 4, 461-476, 1990.
[15] Ajith Abraham, Meta-Learning Evolutionary Artificial Neural Networks, Neurocomputing Journal, Elsevier Science, Netherlands, Vol. 56c, pp. 1-38, 2004.
[16] X. Yao (1993), "A review of evolutionary artificial neural networks", International Journal of Intelligent Systems, 8(4):539- 567.
[17] R. A. Araujo, G. C. Vasconselos and A. E. Ferreria. Hybrid differential evolutionary system for financial time series forecasting. 2007.
[18] H. M. Abdul-Kader. Neural networks training based on differential evolution algorithm compared with other architectures for weather forecasting34. 2009.
[19] Fogel, D. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. Wiley-IEEE Press, 1998.
[20] G. Cybenko. Approximation by superposition of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 303- 314, 1989.
[21] Prof. Dr. Andreas Zell,, WSI Computer Science Department, Computer Arquitecture, Software, Artificial Neural Networks http://www-ra.informatik.uni-tuebingen.de/SNNS/
[22] Storn, Rainer, and Kenneth Price. Differential Evolution –A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization11, 1997, pp. 341-359.
[23] Wickramasinghe, W. and Li, X. (2008), "Choosing Leaders for Multi-objective PSO Algorithms using Differential Evolution", in Proceedings of the seventh International Conference on Simulated Evolution and Learning (SEAL'08), Lecture Notes in Computer Science (LNCS 5361), Springer, p.249 - 258.
[24] Hyndman, R.J. (n.d.) Time Series Data Library, http://www.robjhyndman.com/TSDL. Accessed on February 1st 2010.
[25] Automatic Forecasting Systems. http://www.autobox.com.
[26] Khaing Shwe Wutyi, "Optimization of Profit for Liquefied Petroleum Gas Extraction Plant by using Simplex and Genetic Algorithm", ICCA 2011